- The `while true` condition that the `while` loop examines is the same as you find in an `if` comparison. The same symbols used there are used with `while`, including `==`, `<`, `>`, and `!=`. You can even use the logical dojabbies `&&` (`AND`) or `||` (`OR`), to evaluate multiple comparisons.
- When you apply the language of the `while` loop's format to HEY.C, you get this:

Starting	Line 7	`i=1`
While_true	Line 8	`i<6`
Do_this	Line 11	`i++`

- Notice that, like the `for` keyword, `while` lacks a proper semicolon. It's just followed with a group of statements in curly braces.
- If the `while` loop contains only one statement, you can do without the curly braces:

```
starting;
while(while_true)
      do_this;
```

- Some `while` loops can even be constructed to have no statements. This is rather common:

```
while((do_this)==TRUE)
    ;
```

  In this example, the semicolon is the "statement" that belongs to the `while` loop.

Deciding between a `while` loop and a `for` loop

A `while` loop and a `for` loop have many similarities — so much so that you would almost seem kooky to use `while` if you're fond of `for` and vice versa. As an example, here's the basic `for` loop from the program 100.C, over in Chapter 15:

```
for(i=1;i<=100;i=i+1)
      printf("%d\t",i);
```

This loop counts from 1 to 100 and displays each number.